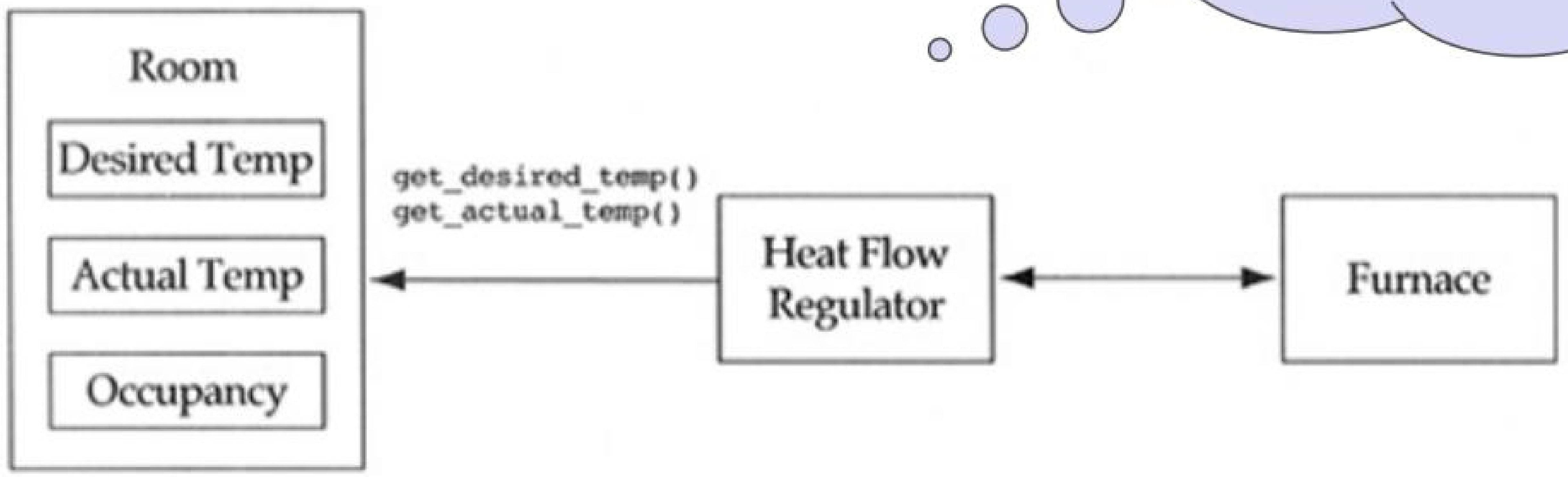
Self Check Exercise: Heat Flow Calculator

Look at the following example: The class *Room* „knows" what its actual and desired temperatures are. At regular intervals, the class *HeatFlowRegulator* calls the methods *get_desired_temp* and *get_actual_temp* and compares the the results. If the desired temperature is higher than the actual temperature, *HeatFlowRegulator* calls the class *Furnace* to heat up the room.

Do you see any weaknesses in this architecture?



Source (diagram): Arthur J. Riel, Object-Oriented Design Heuristics, Addison Wesley, 1996

ExID: ArchitecturalQuality09

Making a Dependency Smaller

The class *HeatFlowRegulator* is too closely coupled to the class Room. For turning the furnace on and off, it is sufficient to know whether the desired temperature is above the actual temperature or not.

This creates a dependency problem: If for instance the format in which temperatures are represented is changed, *HeatFlowRegulator* must change along with it even though it wouldn‘t have to.

The solution is to have *Room* compare the two temperatures. *HeatFlowRegulator* just gets the result. Instead of the two methods on the previous slides, we introduce a new, smaller method called *do_you_need_heat*. It just returns a Boolean value.

For an illustration, see the next slide.